International Conference
ENGINEERING MECHANICS 2010
Svratka, Czech Republic, May 10 – 13, 2010

GENETIC PROGRAMMING APPROXIMATION IN CEMENT PASTE EXPERIMENTAL PERFORMANCE

M. Valtrová, M. Lepš [1]

Summary: *This article introduces results of Genetic Programming used for creation of experimental data approximations together with the search for significant parameters of an affinity cement paste hydration model. Within Genetic Programming trees, the placements of constants still has not been satisfactorily solved. Therefore, the proposed contribution also presents a search for real-valued constants employing Ordinary Least Squares. Twenty trees as results of twenty independent runs of Genetic Programming are presented. From these results the best seven trees are chosen according to specific criteria and the approximations of experimental data are shown. Still, many aspects of Genetic Programming-based symbolic regression are uncovered and especially suppression of the overfitting issues remains unsolved.*

1. Introduction to Genetic Programming

Genetic Programming (GP) is a relatively new form of artificial intelligence and is inspired by Darwinian biological evolution and genetics. GP is an extension of Genetic Algorithms (GA) (Yeun et al., 2005). Contrary to GA that uses string of numbers to represent the solution, Genetic Programming deals with tree-structured program (tree) as an individual (see[2] Figure 1). GP searches highly fit computer programs in the space of all possible programs that solve a problem. The trees are compound of nodes that are elements either from a functional set or from a terminal set. Generally, the functional set consists of mathematical operators, for example $\{+, -, *, /\}$ while the terminal set contains variables or constants. The main difference in the functional set and the terminal set is that the terminal set cannot have arguments.

The evolution of programs is performed through the action of genetic operators and the evaluation of the fitness function (Alvarez et al., 2000). Generally, there are three main genetic operators: reproduction, crossover and mutation. Reproduction is the process of copying individuals according to their fitness value (Yeun et al., 2005). Trees with fitness lower than the average are killed and the new population is filled with the surviving trees. Meanwhile, the crossover is the process of combining information from two trees that are selected from the whole population. Here, one randomly selected subtree is interchanged with another one and two new offsprings are created (see[3] Figure 2). The last genetic operator is mutation (see again

[1] Ing. Martina Valtrová, Ing. Matěj Lepš, Ph.D., Department of Mechanics, Faculty of Civil Engineering, CTU in Prague, Thákurova 7; 166 29 Praha 6; tel: +420 224 355 326, fax: +420 224 310775; e-mail: leps@cml.fsv.cvut.cz
[2] Reproduced from `http://en.wikipedia.org/wiki/Genetic_programming`
[3] Reproduced from `http://www.alesdar.org/oldSite/IS/chap6-3.html`

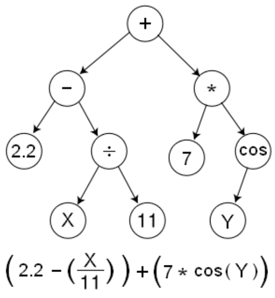


Figure 1: The example of a tree.

Figure 2). Mutation replaces the subtree of selected individuals with new randomly generated subtrees (Yeun et al., 2005). These operations with trees are repeated after several generations until the tree with optimum fitness value is obtained. The principle of genetic programming can be seen in Figure 3.

1.1. Symbolic regression in Genetic Programming

Problems of symbolic regression require finding a function in a symbolic form that fits a given finite sampling of data points (Koza, 1998). One of the main challenges in symbolic regression using Genetic Programming is the handling of constants (or real numbers). There are three known possibilities how to solve it. Koza in (Koza, 1998) has expanded the terminal set by adding new special terminal called the *ephemeral random constant*. Whenever the ephemeral random constant is chosen for any endpoint of the tree during the creation of the initial random population in generation 0, a random number of a specified data type in a specified range is generated and attached to the tree at that point (Koza, 1998). These constants remain fixed for all generations.

The second access to solving constants in symbolic regression is presented in paper (Alvarez et al., 2000). The authors used Genetic Programming for the creation of approximation functions obtained by the response surface methodology. In this work the constants are called *tuning parameters*. They are allocated to a subtree depending on the type of the current node and the structure of the subtree according to the algorithm described in (Alvarez et al., 2000). Once the

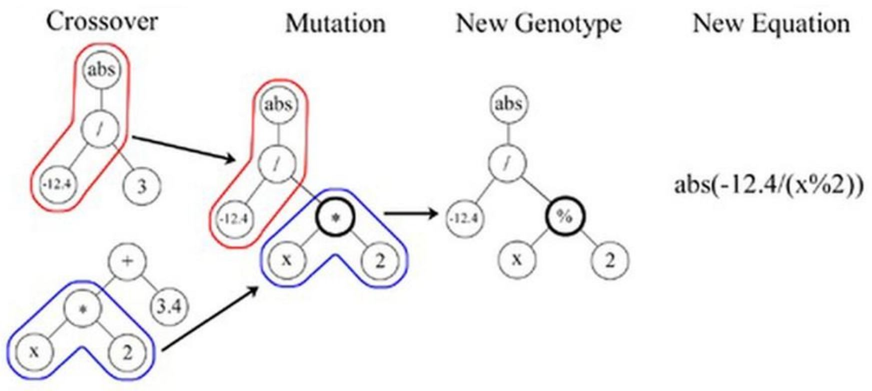


Figure 2: Two genetic operators: the crossover and the mutation.

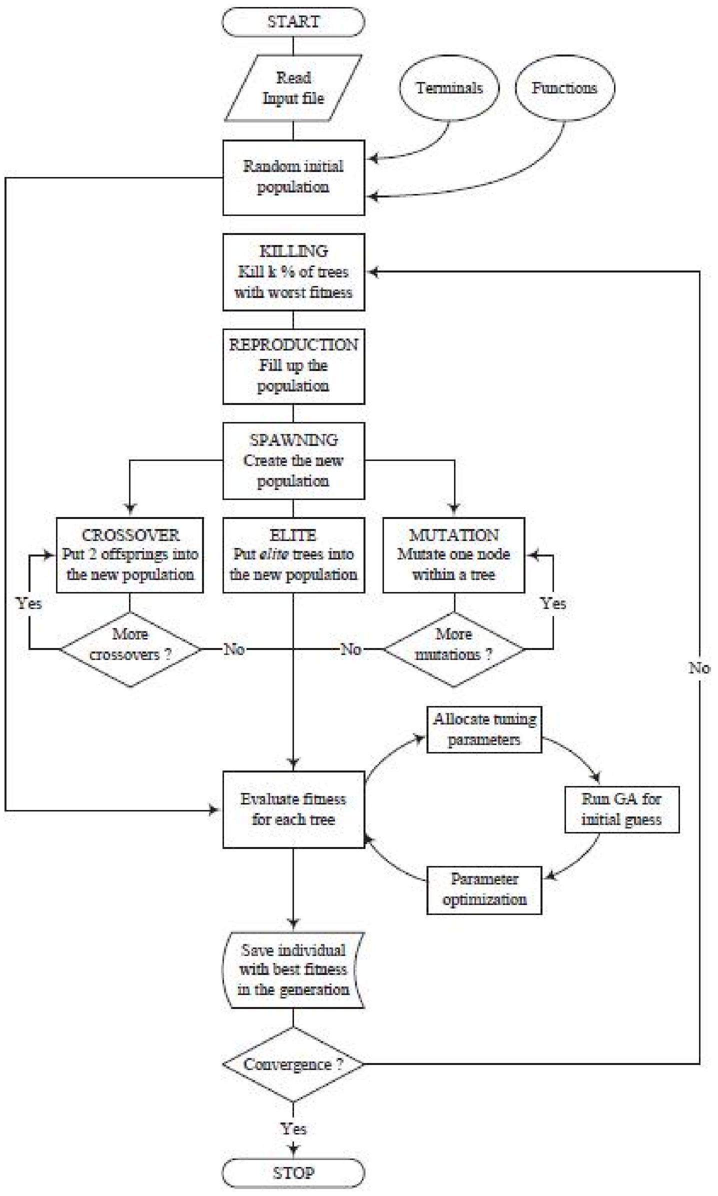

START
Read Input file
Terminals
Functions
Random initial population
KILLING Kill k % of trees with worst fitness
REPRODUCTION Fill up the population
SPAWNING Create the new population
CROSSOVER Put 2 offsprings into the new population
ELITE Put *elite* trees into the new population
MUTATION Mutate one node within a tree
Yes
More crossovers ?
No
No
More mutations ?
Yes
No
Allocate tuning parameters
Run GA for initial guess
Parameter optimization
Evaluate fitness for each tree
Save individual with best fitness in the generation
Convergence ?
Yes
STOP

Figure 3: The flowchart of GP metodology reproduced from (Alvarez et al., 2000).

tuning parameters are allocated at different parts of a tree, a nonlinear optimization method is used to compute them.

The last approach of constants' handling has been used, e.g., in (Yeun et al., 2005). Authors have reported their work regarding GP with polynomials which are applied to fitting a given response surface. They transformed the GP tree into the standard mathematical form by the help of their own translation algorithm. Then the classical regression matrix is created and the ordinary least squares method (OLS) is used to estimate the coefficients. This methodology seems to reduce computational cost in comparison with nonlinear optimization method.

There are more articles concerning symbolic regression in Genetic Programming. For instance, a paper by (Rodríguez-Vázquez and Oliver-Morales, 2004) presents the analysis of the effect of Multi-Branches Genetic Programming in function approximation problems (i.e. symbolic regression problems). Another article published by (Streeter and Becker, 2003) describes the usage of Genetic Programming to automate the discovery of numerical approximation formulae. Nevertheless, neither paper specifies the way how they deal with constants within GP trees.

2. Application of GP to cement paste hydration data

Modeling of hydrating concrete represents a challenging task especially due to multiscale nature and missing mathematical formulation of several underlying phenomena. Missing physical and mathematical description can be replaced by cellular automata model (Šmilauer, 2006). Question arises, whether results from such virtual models can be trusted. In following, Genetic Programming is applied at prediction of hydration heat solely from experimental datasets.

The experimental data was obtained via isothermal calorimetry. Typically, a cement paste is poured in the vial inside temperature-stabilized calorimeter environment (Šmilauer, 2010). The released hydration heat at different times was obtained for the eight samples of cement pastes, see Table 1, (consecutively from top): Data measured at CTU by TAMAir isothermal calorimeter, from (Princigallo et al., 2003), data from private communication and determined from evaporable water content and assumed potential hydration heat 480 J/g, from (Hua et al., 1995), (Robeyst et al., 2007), data measured at CTU by TAMAir isothermal calorimeter and finally, (Tamtsia et al., 2004) assuming potential heat 500 J/g. The released heat depends on seven parameters that include the composition of clinker minerals, gypsum, fineness and water to cement ratio (w/c), hereafter denoted as **X1** – **X7**.

2.1. Modification of experimental data

As was mentioned above, the experimental data was measured at different times. It also means that each sample has a different number of released heat measurements. For example, the cement paste Mokra has 23 719 measurements with the average 0.01 hour time step, whereas Tamtsia has only 5 measurements with the average 10.5 hour time step. For these reasons, the linear interpolation (in some cases extrapolation) has been made to obtain released heat for all cement paste samples at 0 - 600 hours with one hour steps. Hence we achieved same number of measurements for all samples at same times.

Cement paste	**X1** C_3S	**X2** C_2S	**X3** C_3A	**X4** C_4AF	**X5** **Gypsum**	**X6** w/c	**X7** **Fineness**
	Mass %	Mass %	Mass %	Mass %	Vol %	–	m^2/kg
Aalborg	0.6660	0.2380	0.0340	0.0040	0.0360	0.400	390
Principallo	0.6688	0.1131	0.0616	0.0989	0.0510	0.375	530
BAM(Fontana)	0.5491	0.2239	0.0768	0.0712	0.0249	0.300	350
BoumizB35	0.6379	0.0894	0.0816	0.0614	0.0500	0.524	350
Hua	0.6880	0.0750	0.0810	0.0920	0.0400	0.420	400
Robeyst	0.6713	0.0664	0.0590	0.1068	0.0500	0.500	390
Mokra	0.6435	0.1181	0.0399	0.1187	0.0500	0.500	306
Tamtsia	0.5949	0.1603	0.0909	0.0956	0.0500	0.500	340

Table 1: Cement paste samples and appropriate parameters values.

2.2. Affinity hydration models

An affinity model provides a simple framework describing all stages of cement hydration. The rate of hydration can be expressed by temperature-independent *normalized chemical affinity* $\tilde{A}(\alpha)$

$$\frac{\mathrm{d}\alpha}{\mathrm{d}t} = \tilde{A}(\alpha)\exp\left(-\frac{E_a}{RT}\right), \tag{1}$$

where T is an arbitrary constant temperature of hydration, R is the universal gas constant (8.314 Jmol^{-1}K^{-1}) and E_a is the apparent activation energy.

The affinity can be obtained experimentally easily from calorimetry. Isothermal calorimetry measures a heat flow $q(t)$ from a sample and quantifies, after an integration, the hydration heat $Q(t)$. Recognizing $Q(t)/Q_{pot}$ as degree of hydration DoH leads to an approximation which has been slightly modified in (Šmilauer, 2010)

$$\tilde{A}(\alpha) = B_1\left(\frac{B_2}{\alpha_\infty} + \alpha\right)(\alpha_\infty - \alpha)\exp\left(-\bar{\eta}\frac{\alpha}{\alpha_\infty}\right) \tag{2}$$

where $B1, B2$ are coefficients to be calibrated, α_∞ is an ultimate hydration degree and $\bar{\eta}$ represents a microdiffusion of free water through formed hydrates. The parameters for our computations were fitted to $B_1 = 1.0 \cdot 10^7$ 1/h, $B_2 = 2.0 \cdot 10^{-4}$, $\alpha_\infty = 0.9$ and $\bar{\eta} = 7.6$. Since these constants are used for all possible cement pastes, a general curve of DoH development has been obtained by numerical integration of Equation 2 with one hour step producing the same data spacing corresponding to interpolated experimental data. A potential hydration heat Q_{pot} can be obtained from the portland cements' mineral composition

$$Q_{pot}\,[\mathrm{J}] = 517\mathrm{m}_{\mathrm{C_3S}} + 262\mathrm{m}_{\mathrm{C_2S}} + 1144\mathrm{m}_{\mathrm{C_3A}} + 725\mathrm{m}_{\mathrm{C_4AF}} \tag{3}$$

with the masses in grams.

Although the value of potential heat is uniquely given by the amount of the first constituents (X1-X4), see Equation 3, it cannot be directly used with the equation of the DoH development (2) because of unknown B_1, B_2 coefficients for particular cement paste[4]. Therefore, we have

[4] B_1, B_2 coefficients are somehow dependent on all input parameters, however, these relationships are unknown and the search for them exceeds the scope of this work.

tried to find a general expression of Q_{pot} with an addition of all input parameters, i.e. we search for Q_{pot} in

$$Q(\mathbf{X}, t) = Q_{pot}(\mathbf{X}) \cdot DoH(t) \tag{4}$$

that best fits the given data. Note that Q_{pot} is independent on time, which is included through DoH. Moreover, the goal of GP application was not only to create the approximation of experimental data, but also to find out which parameters of cement paste are significant.

2.3. Application of GP

We have used a free GPLAB - a Genetic Programming Toolbox for MATLAB by Sara Silva (for more details see `http://gplab.sourceforge.net/`). Since the GPLAB toolbox does not include a solution of real numbers in symbolic regression in GP, the fist step was to solve this problem. In Section 1.1. we have presented three possibilities how to treat real numbers. After considering all circumstances (e.g. the way in which the GPLAB is programmed), we decided to transform the GP tree into the standard mathematical form and to apply ordinary least squares approach. Note that after the last step a GP solution is actually a polynomial with real-valued coefficients that is used for evaluating the fitness of the corresponding tree.

The GPLAB offers many settings of Genetic Programming. Table 2 shows available options from which several combinations have been tested. In the same table, the final setting that has produced best performance in short runs is presented. Individual terms are explained in GPLAB Manual (Silva, 2007).

Name	**Available Options**	**Chosen Setting**
Genetic Operators	Crossover Mutation Shrink Mutation Swap Mutation	Crossover Mutation
Initialization Methods	Full Grow Ramped Half-and-Half	Ramped Half-and-Half
Expected Number of Offspring	Absolute Rank85 Rank89	Rank85
Sampling Methods	Roulette SUS Tournament Lexictour Doubletour	Lexictour
Elitism	Replace Keep Best Half Elitism Total Elitism	Replace
Survival of The Individuals	Fixed Popsize Resources Pivotfixe	Fixed Popsize

Table 2: Available options of Genetic Programming and chosen setting.

#	Tree	$\mathbf{R^2}$
1	$[\mathrm{X5}*\mathrm{X5} \quad -\mathrm{X1}*\mathrm{X5} \quad \mathrm{X1}*\mathrm{X3} \quad -\mathrm{X3}*\mathrm{X5}]$	0.833
2	$[\mathrm{X6}*\mathrm{X6}*\mathrm{X5} \quad -\mathrm{X6}*\mathrm{X5}*\mathrm{X5} \quad \mathrm{X6}*\mathrm{X6} \quad -\mathrm{X6}*\mathrm{X5} \quad \mathrm{X3} \quad \mathrm{X7}]$	0.854
3	$[\mathrm{X7}^2*\mathrm{X3}*\mathrm{X4} \quad -\mathrm{X7}^2*\mathrm{X3}*\mathrm{X6} \quad -\mathrm{X7}^3*\mathrm{X4} \quad \mathrm{X7}^3*\mathrm{X6} \quad -\mathrm{X7}^2*\mathrm{X5}*\mathrm{X4} \quad \mathrm{X7}^2*\mathrm{X6}*\mathrm{X5}]$	0.567
4	$[3*\mathrm{X5}*\mathrm{X4X3}*\mathrm{X4} \quad -\mathrm{X7}*\mathrm{X4} \quad -3*\mathrm{X6}*\mathrm{X5} \quad -\mathrm{X3}*\mathrm{X6} \quad \mathrm{X7}*\mathrm{X6}]$	0.813
5	$[\mathrm{X5}^2 \quad \mathrm{X6} \quad \mathrm{X4} \quad \mathrm{X5} \quad -\mathrm{X3} \quad \mathrm{X7}]$	0.892
6	$[\mathrm{X5} \quad \mathrm{X7} \quad -\mathrm{X3} \quad -\mathrm{X5}*\mathrm{X4} \quad -\mathrm{X5}*\mathrm{X6} \quad -\mathrm{X6}]$	0.889
7	$[\mathrm{X6} \quad \mathrm{X4}^2 \quad 2*\mathrm{X4}*\mathrm{X6} \quad \mathrm{X6}^2 \quad -\mathrm{X3}^2]$	0.890
8	$[\mathrm{X6} \quad \mathrm{X3} \quad -\mathrm{X5} \quad \mathrm{X7} \quad -\mathrm{X5}^2 \quad \mathrm{X4}]$	0.892
9	$[\mathrm{X7}*\mathrm{X5} \quad \mathrm{X7}*\mathrm{X3} \quad -\mathrm{X3}*\mathrm{X1}*\mathrm{X5} \quad -\mathrm{X3}^2*\mathrm{X1} \quad -\mathrm{X4}*\mathrm{X5} \quad -\mathrm{X4}*\mathrm{X3}]$	0.881
10	$[\mathrm{X6}^2*\mathrm{X2} \quad -\mathrm{X6}*\mathrm{X3} \quad -\mathrm{X6}*\mathrm{X2}^2 \quad \mathrm{X2}*\mathrm{X3} \quad -\mathrm{X3}*\mathrm{X6}*\mathrm{X2} \quad \mathrm{X3}^2 \quad \mathrm{X1}*\mathrm{X6}*\mathrm{X2} \quad -\mathrm{X1}*\mathrm{X3}]$	0.895
11	$[2*\mathrm{X5}*\mathrm{X7}^2 \quad \mathrm{X7}*\mathrm{X5}^2 \quad -\mathrm{X1}*\mathrm{X7}^2 \quad -\mathrm{X7}*\mathrm{X1}*\mathrm{X5} \quad \mathrm{X7}^3 \quad -\mathrm{X2}*\mathrm{X7}^2 \quad -\mathrm{X7}*\mathrm{X2}*\mathrm{X5}]$	0.853
12	$[\mathrm{X6}*\mathrm{X2} \quad \mathrm{X6}*\mathrm{X5} \quad -\mathrm{X4}*\mathrm{X2} \quad -\mathrm{X4}*\mathrm{X5} \quad -\mathrm{X2}*\mathrm{X7} \quad -\mathrm{X5}*\mathrm{X7} \quad \mathrm{X3}]$	0.822
13	$[\mathrm{X1}*\mathrm{X7}^2*\mathrm{X4} \quad \mathrm{X1}*\mathrm{X7}*\mathrm{X4} \quad -\mathrm{X1}*\mathrm{X2}*\mathrm{X7}*\mathrm{X4} \quad -\mathrm{X1}*\mathrm{X2}*\mathrm{X4} \quad -\mathrm{X7}^3*\mathrm{X4} \quad -\mathrm{X7}^2*\mathrm{X4} \quad \mathrm{X2}*\mathrm{X7}^2*\mathrm{X4} \quad \mathrm{X2}*\mathrm{X7}*\mathrm{X4}]$	0.327
14	$[\mathrm{X2}*\mathrm{X5}*\mathrm{X7}*\mathrm{X3} \quad -\mathrm{X2}*\mathrm{X7}*\mathrm{X1}*\mathrm{X5} \quad \mathrm{X2}^2*\mathrm{X5}*\mathrm{X3} \quad -\mathrm{X2}^2*\mathrm{X5}*\mathrm{X1} \quad -\mathrm{X2}*\mathrm{X1}*\mathrm{X7}*\mathrm{X3} \quad \mathrm{X2}*\mathrm{X1}^2*\mathrm{X7} \quad -\mathrm{X1}*\mathrm{X2}^2*\mathrm{X3} \quad \mathrm{X1}^2*\mathrm{X2}^2]$	0.895
15	$[\mathrm{X1}*\mathrm{X7} \quad \mathrm{X7}*\mathrm{X6} \quad \mathrm{X5}*\mathrm{X7} \quad \mathrm{X7}*\mathrm{X4} \quad \mathrm{X1}*\mathrm{X5} \quad \mathrm{X6}*\mathrm{X5} \quad \mathrm{X5}^2 \quad \mathrm{X4}*\mathrm{X5}]$	0.895
16	$[\mathrm{X7}*\mathrm{X6} \quad \mathrm{X6}*\mathrm{X4} \quad -\mathrm{X6}*\mathrm{X5} \quad -\mathrm{X4}^2 \quad \mathrm{X4}*\mathrm{X5} \quad \mathrm{X7}^2 \quad -\mathrm{X5}*\mathrm{X7}]$	0.801
17	$[\mathrm{X3}^2 \quad \mathrm{X3}*\mathrm{X6} \quad -\mathrm{X3}*\mathrm{X7} \quad -\mathrm{X3}*\mathrm{X1} \quad \mathrm{X5}*\mathrm{X3} \quad \mathrm{X6}*\mathrm{X5} \quad -\mathrm{X5}*\mathrm{X7} \quad -\mathrm{X1}*\mathrm{X5}]$	0.895
18	$[2*\mathrm{X6}^2*\mathrm{X4} \quad \mathrm{X6}*\mathrm{X4}*\mathrm{X7} \quad -2*\mathrm{X6}^2*\mathrm{X5} \quad -\mathrm{X6}*\mathrm{X5}*\mathrm{X7} \quad -2*\mathrm{X3}*\mathrm{X4}*\mathrm{X6} \quad -\mathrm{X3}*\mathrm{X4}*\mathrm{X7} \quad 2*\mathrm{X5}*\mathrm{X3}*\mathrm{X6} \quad \mathrm{X5}*\mathrm{X7}*\mathrm{X3}]$	0.895
19	$[\mathrm{X7}*\mathrm{X6}^2*\mathrm{X1} \quad \mathrm{X7}*\mathrm{X6}^3 \quad \mathrm{X6}^2*\mathrm{X5}*\mathrm{X1} \quad \mathrm{X6}^3*\mathrm{X5} \quad -\mathrm{X6}*\mathrm{X1}*\mathrm{X7}^2 \quad -\mathrm{X7}^2*\mathrm{X6}^2 \quad -\mathrm{X6}*\mathrm{X7}*\mathrm{X1}*\mathrm{X5} \quad -\mathrm{X6}^2*\mathrm{X5}*\mathrm{X7}]$	0.895
20	$[\mathrm{X5}*\mathrm{X7}*\mathrm{X6} \quad -\mathrm{X5}^2*\mathrm{X7} \quad \mathrm{X5}*\mathrm{X1}*\mathrm{X6} \quad -\mathrm{X5}^2*\mathrm{X1} \quad \mathrm{X4}*\mathrm{X7}*\mathrm{X6} \quad -\mathrm{X4}*\mathrm{X7}*\mathrm{X5} \quad \mathrm{X4}*\mathrm{X1}*\mathrm{X6} \quad -\mathrm{X4}*\mathrm{X1}*\mathrm{X5}]$	0.895

Table 3: Best trees out of 20 independent runs (without regression coefficients) with appropriate $\mathbf{R^2}$ coefficients ($\mathbf{R^2} = 1$ means perfect fit).

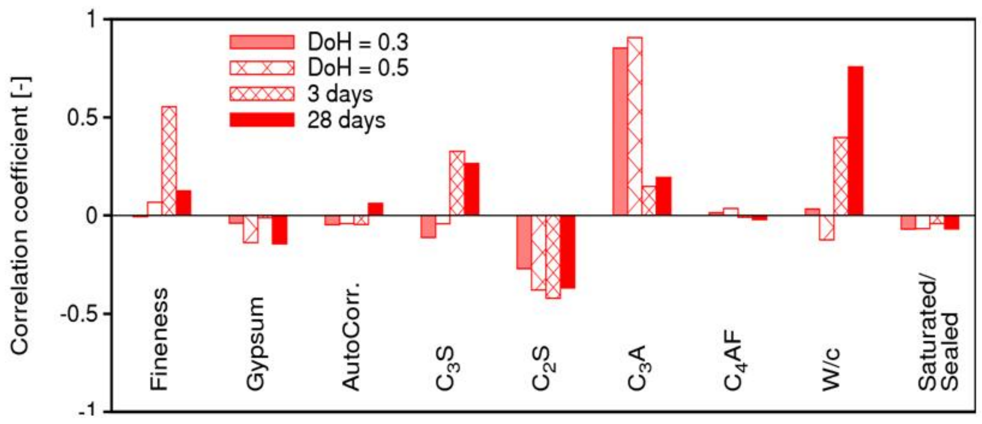


Figure 4: Correlation between input parameters and hydration heat.

The function set contained mathematical operators $\{+, -, *\}$, the terminal set consisted of variables only. To avoid a bloat, a phenomenon consisting of an excessive code growth without the corresponding improvement in fitness (Silva, 2007), the maximum of 14 nodes was set. Finally, the population size of 500 individuals and the maximum number of generation of 100 was used.

2.4. Results

One optimization run was repeated 20 times for the sake of statistical relevance. In the last ten runs we have increased the maximum number of generation to 150. Table 3 presents best trees (without regression coefficients) along with appropriate coefficients of determination $\mathbf{R^2}$ (the more coefficient of determination is closer to one the better a model is).

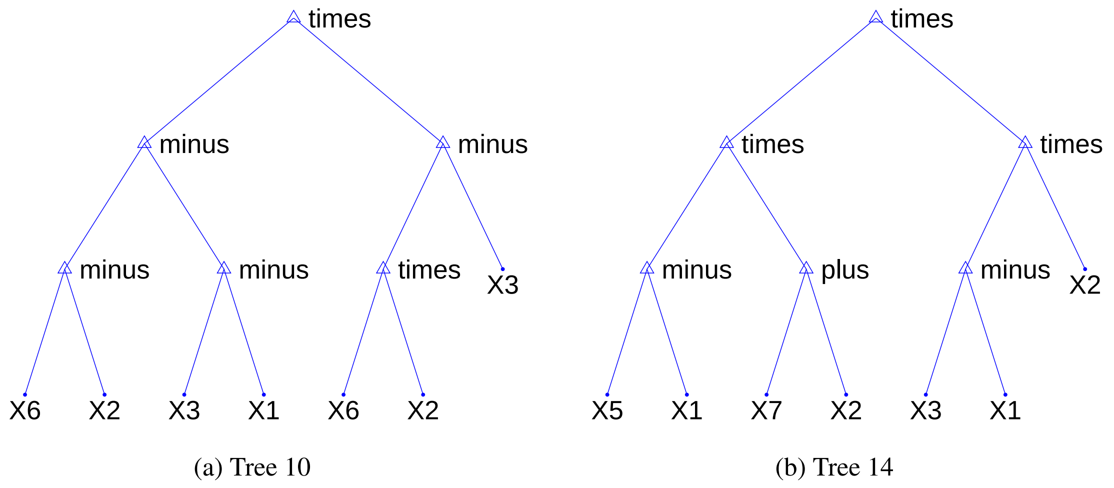
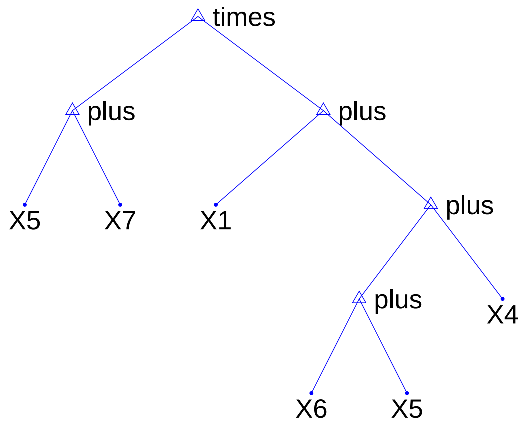
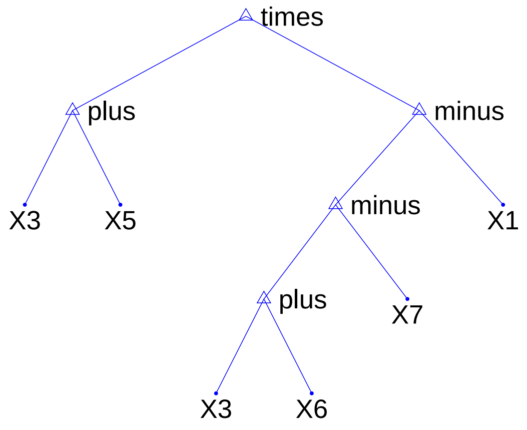


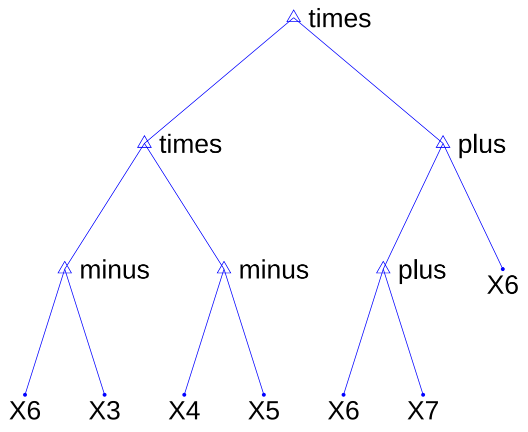
Figure 5: Chosen trees. Another trees on the next page.



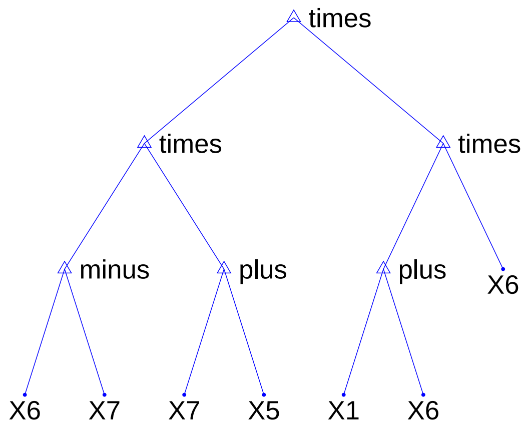
(c) Tree 15



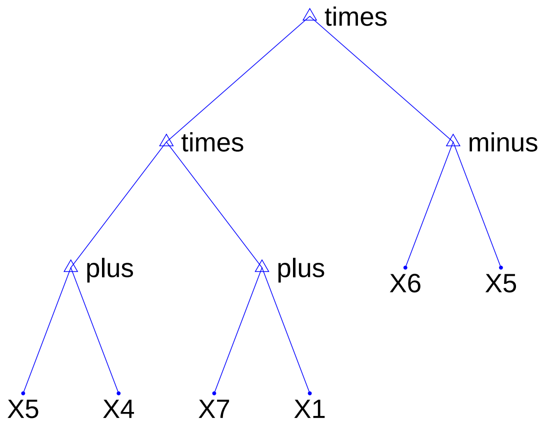
(d) Tree 17



(e) Tree 18



(f) Tree 19



(g) Tree 20

Figure 5: Chosen trees.

Table 4: Coefficients of chosen trees.

Tree Number	**Figure**	**Coefficients**
10	Figure 5 (a)	$-8.111895608905244e+003$
		$-1.331146211137479e+004$
		$2.166991906154698e+004$
		$2.094929318789972e+004$
		$3.146359656923416e+004$
		$-5.159089486510225e+004$
		$1.656638220837784e+004$
		$-2.502062966745425e+003$
14	Figure 5 (b)	$9.243869108120686e+005$
		$1.092885651009214e+005$
		$-1.642636544079870e+009$
		$-1.958372097860084e+008$
		$6.851121570865510e+004$
		$8.144787171264200e+003$
		$-1.187129296385552e+008$
		$-1.420841261157905e+007$
15	Figure 5 (c)	$1.458193574558148e+001$
		$-3.766630462198184e+001$
		$2.360524887640847e+002$
		$-1.259300390804292e+001$
		$-1.408600541438428e+005$
		$4.206396918009609e+005$
		$-2.631407720512701e+006$
		$1.420785625906093e+005$
17	Figure 5 (d)	$-2.466383416090238e+005$
		$1.655046930775791e+005$
		$-3.180135520602344e+002$
		$2.394740423354011e+005$
		$-8.621761087793748e+004$
		$-1.899630841849877e+005$
		$3.756803751370741e+002$
		$-3.274857350250676e+005$
18	Figure 5 (e)	$9.218232129081906e+004$
		$-2.625798901773804e+002$
		$1.536607745026151e+005$
		$-4.779718445863055e+002$
		$6.730594812736753e+005$
		$-1.929012856373917e+003$
		$1.210291943991558e+006$
		$-3.322982616715128e+003$
19	Figure 5 (f)	$-6.073079781048356e+001$
		$1.827081619924882e+002$
		$1.038139056753326e+006$
		$-2.105423533983755e+006$
		$-3.724875411907906e-001$
		$6.622967088874029e-001$
		$3.459207518943232e+003$
		$-5.916622733643175e+003$
20	Figure 5 (g)	$-1.125412110266874e+003$
		$-3.335362469264811e+004$
		$-1.426079068040327e+005$
		$1.005745437237222e+007$
		$6.555383350476078e+002$
		$1.636276852918849e+004$
		$1.622251159086200e+005$
		$-3.585797942507672e+006$

Next, we can estimate the effect of input parameters on the hydration heat from sensitivity of the chemical model CEMHYD3D, see Figure 4. CEMHYD3D is a 3D portland cement

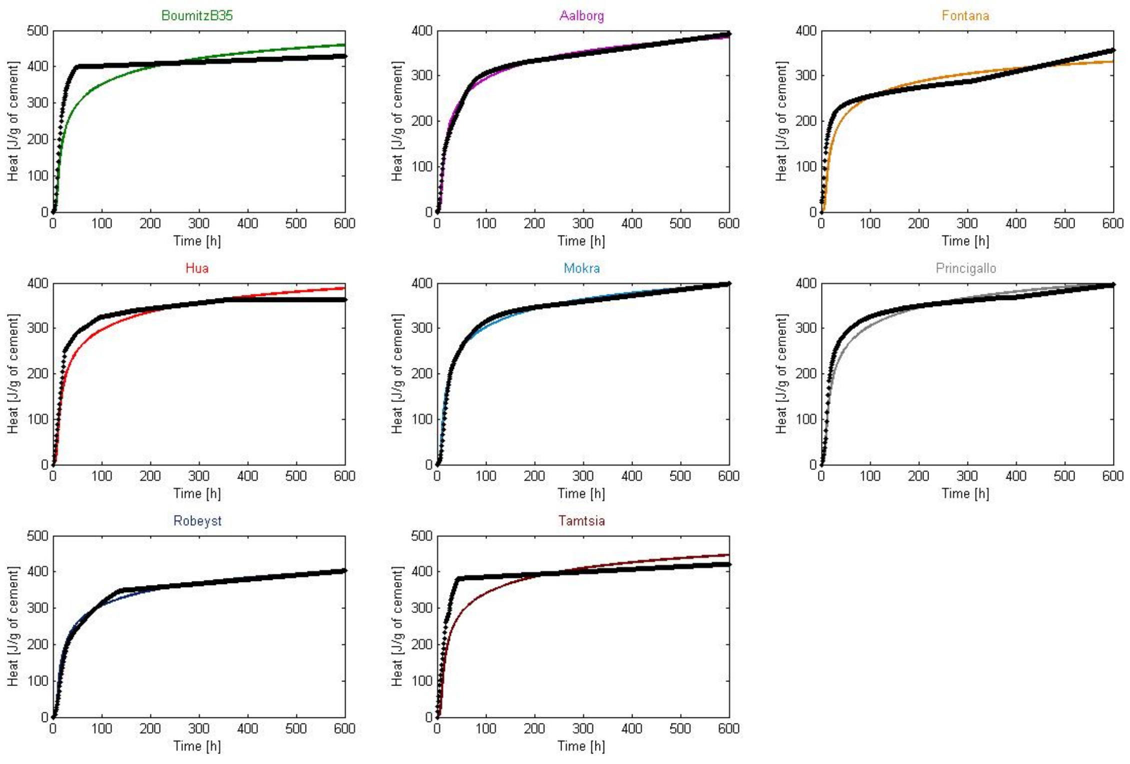


Figure 6: Approximation of experimental data.

hydration model based on cellular automata in combination with basic chemical reactions. For more details about CEMHYD3D see, e.g. (Šmilauer, 2006). We have chosen seven best trees according to sensitivities and coefficients of determination, see again Table 3. The best trees are plotted in Figure 5 and appropriate coefficients are shown in Table 4.

We also created approximation of experimental data as shown in Figure 6. We present only the tree 10 approximation of experimental data because other approximations are almost the same and the differences are not visible.

3. Conclusions

The goal of this article was (i) to create an approximation of experimental data and (ii) to find out which parameters of cement paste are significant. The fitted response surfaces are almost perfect, however, a detailed examination of results shows that no tree contains all important input variables (X1, X2, X3, X6 and X7). One of the missing variables X1 – X3 can be replaced by X4 variable since X1 – X4 are linearly dependent constituting a volume unity. But still, inclusion of this relationship into GP will surely not bring substantial improvement. Rough enumeration of obtained trees over the allowable domain shows classical signs of overfitting. Therefore, more data are needed or more strict selection mechanism within GP based on the lenght and/or complexity of regression trees have to be employed. Finally, a problem of constants' placement within GP trees has been successfully solved, however, the OLS approach limits dramatically the functions list. Therefore, our next step will be aimed at the ephemeral random constant methodology.

4. Acknowledgment

The financial support of this work by MPO research project FT-TA4/100 is gratefully acknowledged.

5. References

Alvarez, L. F., Toropov, V. V., Hughes, D. C., and Ashour, A. F. (2000). Approximation model building using genetic programming methodology: Application. Department of Civil and Environmental Engineering, University of Bradford, UK.

Hua, C., Acker, P., and Ehrlacher, A. (1995). Analyses and models of the autogenous shrinkage of hardening cement paste. I. modelling at macroscopic scale. *Cement and Concrete Research*, 25(7):1457–1468.

Koza, J. R. (1998). *Genetic Programming*. A Bradford book, sixth edition.

Princigallo, A., Lura, P., van Breugel, K., and Levita, G. (2003). Early development of properties in a cement paste: A numerical and experimental study. *Cement and Concrete Research*, 33(7):1013 – 1020.

Robeyst, N., Gruyaert, E., and Belie, N. D. (2007). *Advances in Construction Materials 2007: Ultrasonic and calorimetric measurements on fresh concrete with blast-furnace slag*, chapter VI, pages 497–504. Springer Berlin Heidelberg.

Rodríguez-Vázquez, K. and Oliver-Morales, C. (2004). Function approximation by means of multi-branches genentic programming. IIMAS-UNAM, Circuito Escolar, Ciudad Universitaria, Mexico.

Silva, S. (2007). *GPLAB Manual*. ECOS - Evolutionary and Complex Systems Group, Portugal, 3^{rd} edition.

Streeter, M. and Becker, L. A. (2003). Automated discovery of numerical approximation formulae via genetic programming. Department of Computer Science, Worcester Polytechnic Institute, Worcester.

Tamtsia, B., Beaudoin, J., and Marchand, J. (2004). The early age short-term creep of hardening cement paste: Load-induced hydration effects. *Cement and Concrete Composites*, 26:481–489.

Šmilauer, V. (2006). *Elastic properties of hydrating cement paste determined from hydration models*. PhD thesis, CTU in Prague. `http://mech.fsv.cvut.cz/~smilauer/`.

Šmilauer, V. (To appear, 2010). *Multiscale Modeling of Hydrating Concrete*. Saxe-Coburg Publications, Stirling.

Yeun, Y. S., Yang, Y. S., Ruy, W. S., and Kim, B. J. (2005). Polynomial genetic programming for response surface modeling. Part 1: A methodology. *Structural and Multidisciplinary Optimization*, 29:19–34.